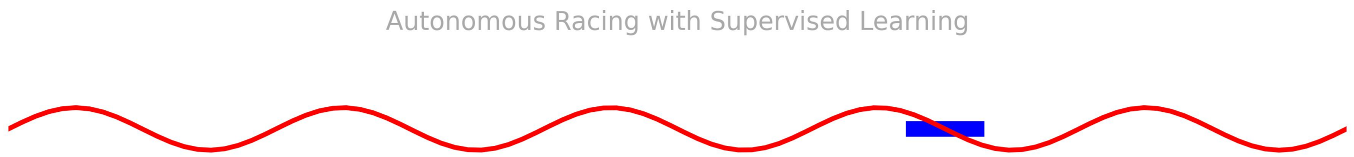
AI Project - TORCS Neural Network Driver

A machine learning-based autonomous driving system for The Open Racing Car Simulator (TORCS) that uses neural networks to learn from human driving data.



Autonomous Racing with Supervised Learning

Group Members

- Ahmed Ali Zahid 22i-1271
- Shaharyar Rizwan 22i-0999
- Abdul Waseh Zafar 22i-0823

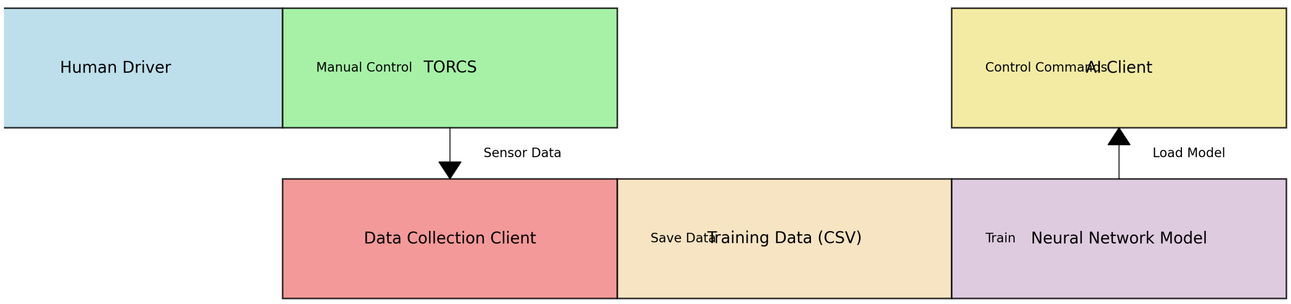
Overview

The implementation consists of three main components:

1. **Data Collection** - A modified pyScrcClient for collecting driving data in manual mode
2. **Model Training** - A neural network trainer that learns from collected driving data
3. **AI Client** - An autonomous client that uses the trained model to drive in TORCS

System Architecture

TORCS Neural Network System Architecture



The system pipeline consists of:

1. **Data Collection Phase**: Human demonstrations are recorded using a modified pyScrcClient
2. **Training Phase**: A neural network learns from the collected data
3. **Inference Phase**: The trained model is used for autonomous driving

Features

- **Supervised Learning**: Learns driving behavior directly from human demonstrations
- **Modular Design**: Separates data collection, model training, and autonomous driving
- **Extensible Architecture**: Easy to modify for different tracks, cars, and model architectures
- **Optimized Performance**: Utilizes AVX/AVX2 instructions for faster neural network computation on compatible CPUs
- **Graceful Fallbacks**: Handles missing dependencies and model components

Requirements

- Python 3.6+
- TORCS with SCR Server patch
- Python packages:
  - TensorFlow 2.x
  - pandas
  - numpy
  - scikit-learn
  - matplotlib
  - joblib

Installation

1. Install TORCS with the SCR server patch
2. Clone this repository
3. Install required Python packages:

```
pip install tensorflow pandas numpy scikit-learn matplotlib joblib
```

Data Collection

The system uses a modified version of the pyScrcClient to record driving data when a human is controlling the car.

Running the Data Collection Client

```
python pyScrcClient/src/client.py --track G-Speedway --car ToyotaCorollaWRC --mode manual
```

Command line arguments:

- `--track` - Track name (G-Speedway, E-Track3, Dirt2)

- `--car` - Car model (ToyotaCorollaWRC, Peugeot406, MitsubishiLancer)
- `--mode` - Driving mode (manual, ai)
- `--run_id` - Numeric ID for the recording session

Controls

In manual mode, the following keyboard controls are used:

- **W/S**: Accelerate/Brake
- **A/D**: Steer left/right
- **Q/E**: Shift down/up
- **Space**: Clutch
- **M**: Toggle between manual and AI mode

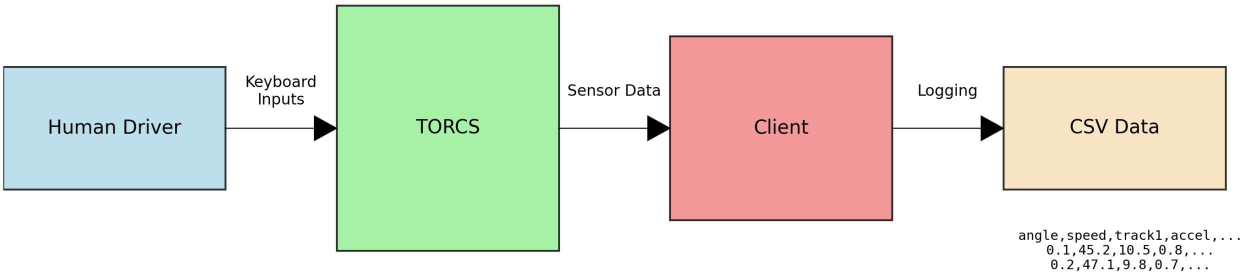
Data Structure

The collected data is saved in CSV format:

```
pyScrcClient/data/[track]/[track]_[car]_[mode]_[run_id].csv
```

Example: `pyScrcClient/data/G-Speedway/G-Speedway_ToyotaCorollaWRC_manual_01.csv`

Data Collection Process



```
angle,speed,track1,accel,...
  0.1,45.2,10.5,0.8,...
   0.2,47.1,9.8,0.7,...
```

Each CSV file contains rows with the following types of data:

- **Sensor readings**: Track edges, speed, wheel velocities, etc.
- **Control outputs**: Steering, acceleration, braking, clutch, gear

Model Training

After collecting driving data, the neural network can be trained using:

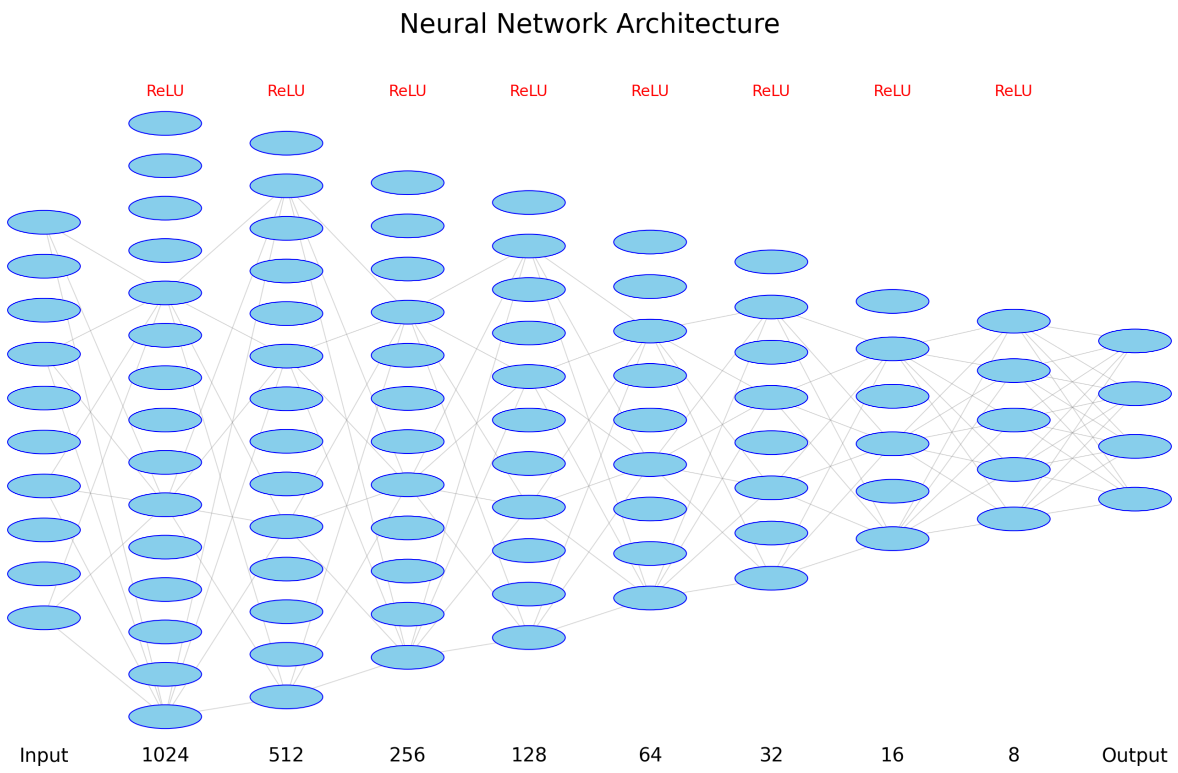
```
python client_model.py
```

The training process:

1. Loads all CSV files from the data directory
2. Preprocesses the data (normalization, cleaning, splitting)
3. Trains a neural network model
4. Saves the model and visualization

Neural Network Architecture

The model uses a feed-forward neural network with the following layers:



```
Input Layer (sensor data) → 1024 → 512 → 256 → 128 → 64 → 32 → 16 → 8 → Output
Layer (controls)
```

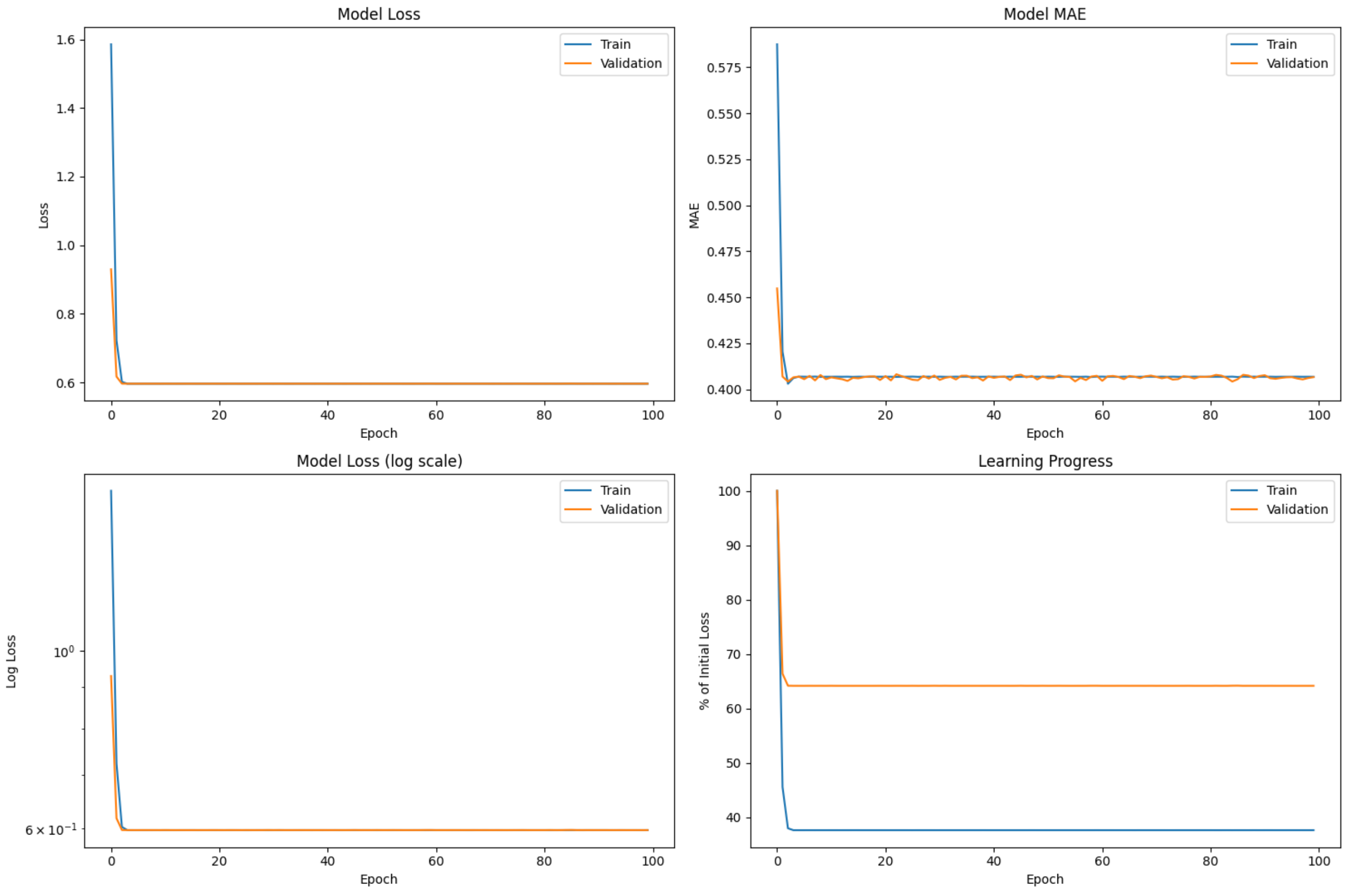
All hidden layers use ReLU activation functions, and the network is trained using the Adam optimizer with mean squared error loss.

Model Output

The training script saves several files to the `model/` directory:

- `torcs_model.h5` - The trained Keras model
- `torcs_model_scaler.pkl` - The fitted StandardScaler for input normalization
- `torcs_model_input_cols.npy` - Input column names

- `torcs_model_output_cols.npy` - Output column names
- `training_history.png` - Training visualization
- `model_summary.txt` - Model architecture details



Autonomous Driving

To run the autonomous driving client using the trained model:

```
python ai_client.py --model model/torcs_model --debug
```

Command line arguments:

- `--model` - Path to the trained model (without .h5 extension)
- `--host` - TORCS server host (default: localhost)
- `--port` - TORCS server port (default: 3001)
- `--logging` - Enable logging of sensor and control data
- `--debug` - Enable debug output

Performance Metrics

The AI client logs performance metrics during runtime, including:

- Average prediction time
- Control values (steering, acceleration, braking)
- Current gear and speed

Implementation Details

pyScrcClient Integration

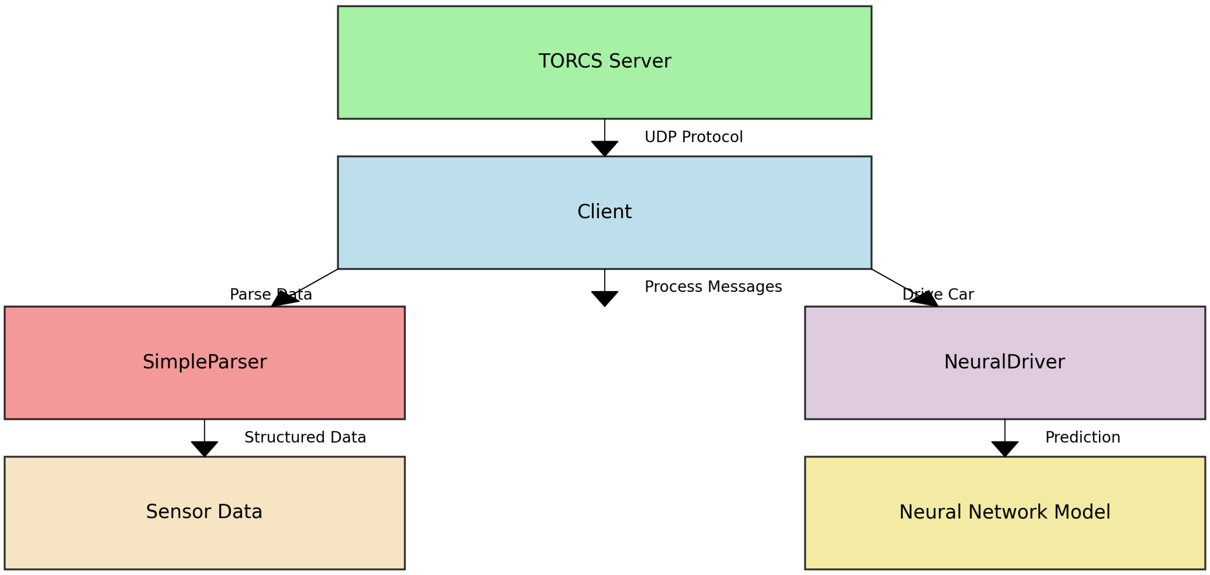
1. Enhanced data logging capabilities
2. Manual driving mode
3. Neural network integration

Client Architecture

The AI client's architecture consists of:

- **SimpleParser**: Converts TORCS sensor data to structured format
- **NeuralDriver**: Uses the neural model to make driving decisions
- **Client**: Handles communication with the TORCS server

TORCS AI Client Architecture



Supported Tracks and Cars

Tracks:

- G-Speedway (Oval track, high speed)
- E-Track3 (Road track, technical)
- Dirt2 (Dirt track, low traction)

Cars:

- ToyotaCorollaWRC
- Peugeot406
- MitsubishiLancer

Project Structure

```
.
├── ai_client.py           # Neural network driver client
├── client_model.py        # Model training script
├── model/                 # Trained models directory
│   ├── torcs_model.h5
│   ├── torcs_model_scaler.pkl
│   └── training_history.png
├── pyScrcClient/          # Modified pyScrcClient
│   ├── data/              # Collected driving data
│   └── src/               # Client source code
│       ├── client.py      # Main client for data collection
│       ├── driver.py      # Driver implementation
│       ├── carState.py    # Car state representation
│       └── msgParser.py   # TORCS protocol parser
└── README.md              # This documentation
```